# Large scale non-linear learning on a single CPU

Andreas Mueller
NYU / scikit-learn

- Large Scale – “Out of core: Fits on a hard disk but in RAM”

- Large Scale – “Out of core: Fits on a hard disk but in RAM”

- Non-linear – because real-world problems are not.

- Large Scale – “Out of core: Fits on a hard disk but in RAM”
- Non-linear – because real-world problems are not.
- Single CPU – Because parallelization is hard (and often unnecessary)

# Three regimes of data

- Fits in RAM
- Fits on a Hard Drive
- Doesn't fit on a single PC

# Three regimes of data

- Fits in RAM (up to 256 GB?)
- Fits on a Hard Drive (up to 6TB?)
- Doesn't fit on a single PC

# Nobody ever got fired for using Hadoop on a cluster

Antony Rowstron, Dushyanth Narayanan, Austin Donnelly, Greg O'Shea, and Andrew Douglas

10 April 2012

- Why not to do out of core learning
- The scikit-learn way
  - Hashing trick
  - Kernel approximation
  - Random neural nets
  - Supervised Feature Extraction
  - Neural nets
- What else is out there

Why not do to out of core learning.

Your data is not that big!

| | vCPU | ECU | Memory (GiB) | Instance Storage (GB) | Linux/UNIX Usage |
|---|---|---|---|---|---|
| **Memory Optimized - Current Generation** | | | | | |
| r3.large | 2 | 6.5 | 15 | 1 x 32 SSD | $0.195 per Hour |
| r3.xlarge | 4 | 13 | 30.5 | 1 x 80 SSD | $0.39 per Hour |
| r3.2xlarge | 8 | 26 | 61 | 1 x 160 SSD | $0.78 per Hour |
| r3.4xlarge | 16 | 52 | 122 | 1 x 320 SSD | $1.56 per Hour |
| r3.8xlarge | 32 | 104 | 244 | 2 x 320 SSD | $3.12 per Hour |
| **Storage Optimized - Current Generation** | | | | | |
| i2.xlarge | 4 | 14 | 30.5 | 1 x 800 SSD | $0.938 per Hour |
| i2.2xlarge | 8 | 27 | 61 | 2 x 800 SSD | $1.876 per Hour |
| i2.4xlarge | 16 | 53 | 122 | 4 x 800 SSD | $3.751 per Hour |
| i2.8xlarge | 32 | 104 | 244 | 8 x 800 SSD | $7.502 per Hour |

"256Gb ought to be enough for anybody."

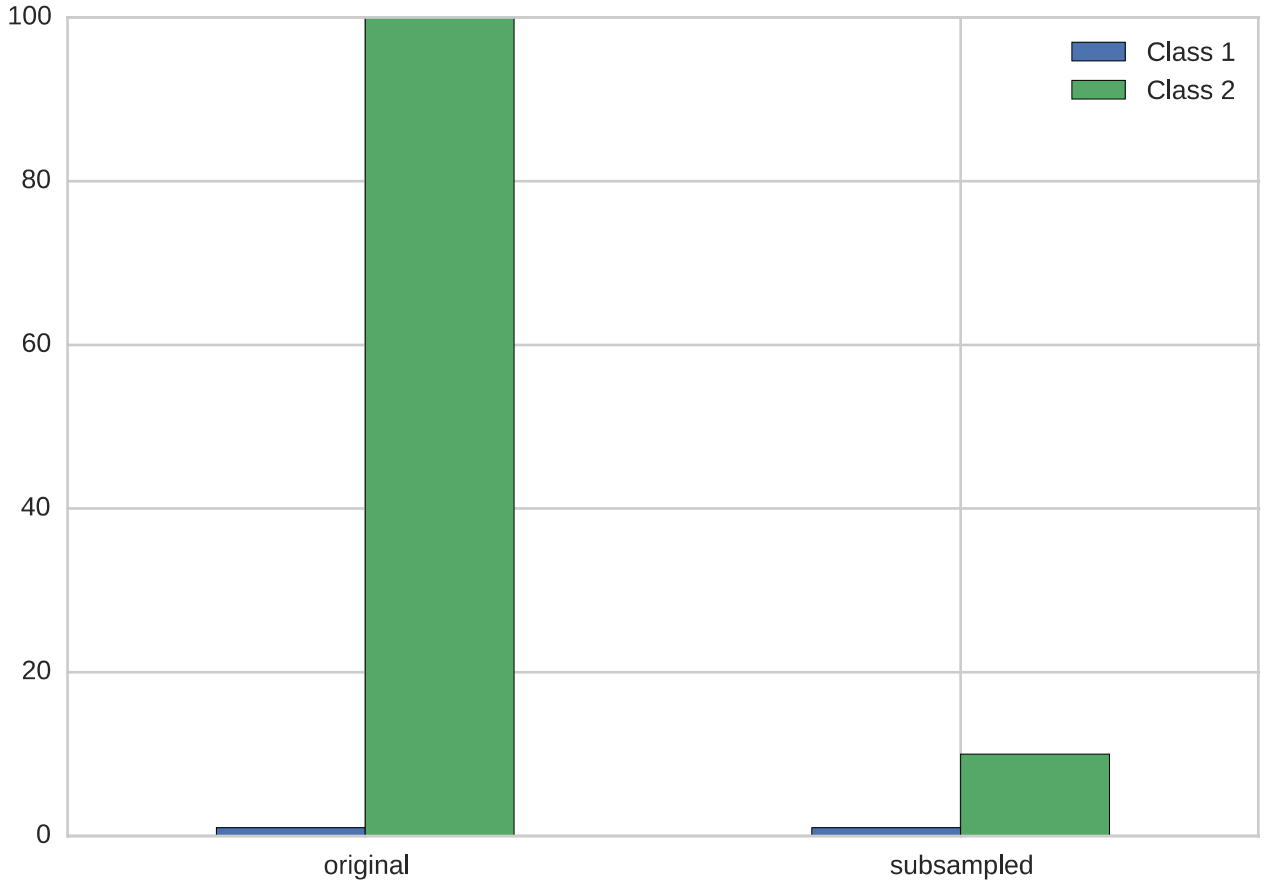
- me

"256Gb ought to be enough for anybody."
- me

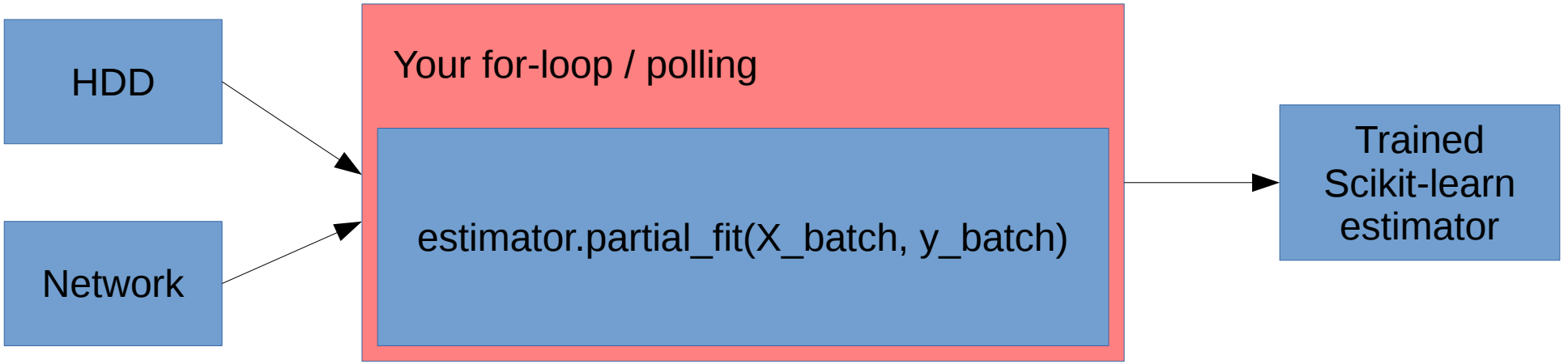
(for machine learning)

# Subsample!

# Subsample!



# The scikit-learn way


HDD
Network
Your for-loop / polling
estimator.partial_fit(X_batch, y_batch)
Trained
Scikit-learn
estimator

# Linear Classification

```
from sklearn.linear_model import SGDClassifier

sgd = SGDClassifier()

for batch_name in glob("*.pickle"):
    with open(batch_name) as f:
        X_batch, y_batch = pickle.load(batch_name)
        sgd.partial_fit(X_batch, y_batch, classes=[0, 1])
```

# Linear Classification

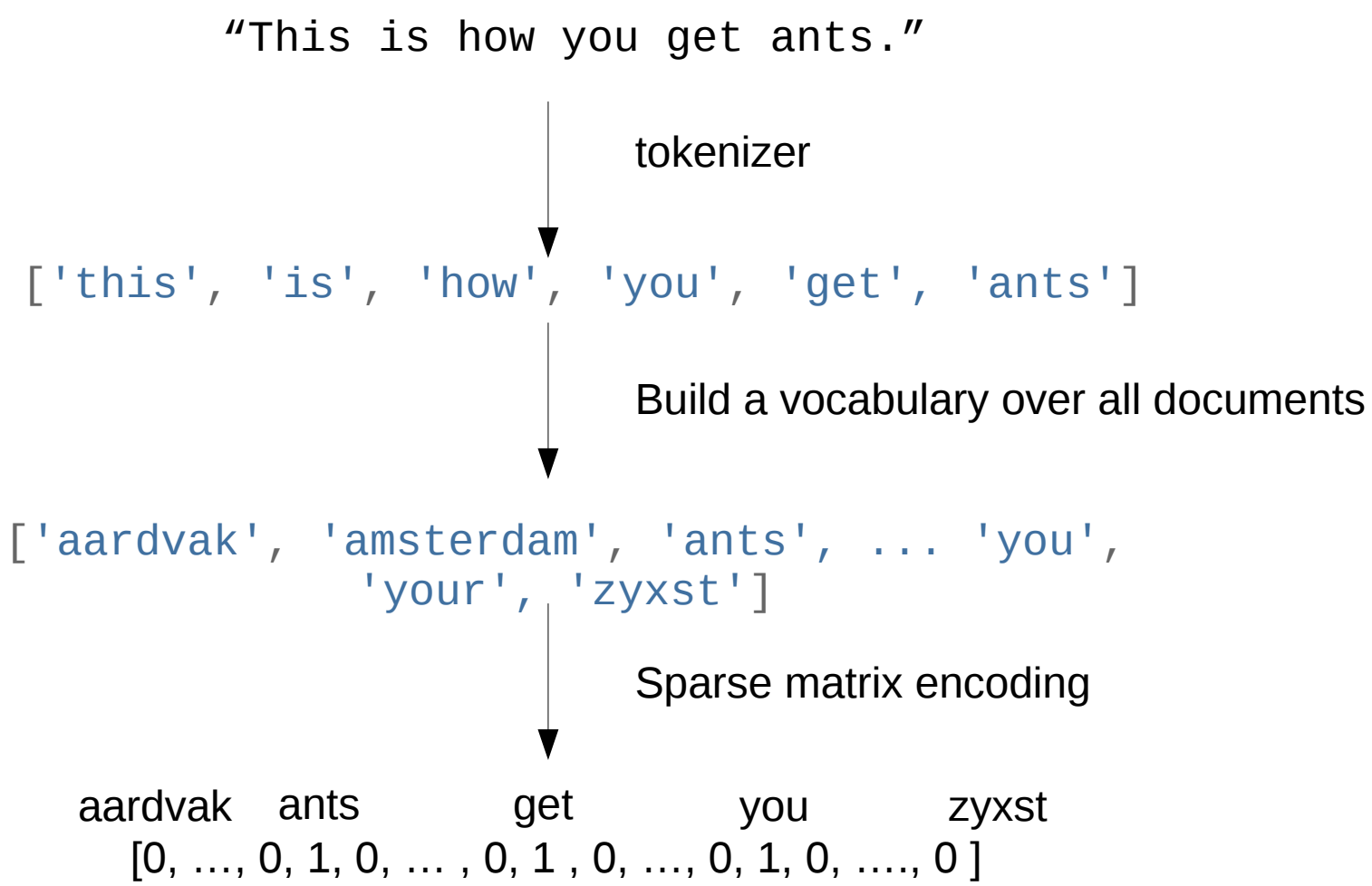
```
from sklearn.linear_model import SGDClassifier

sgd = SGDClassifier()

csv_iterator = pd.read_csv("my_large_file.csv", chunksize=10000)
for chunk in csv_iterator:
    X_batch = csv_iterator[features]
    y_batch = csv_iterator["label"]
    sgd.partial_fit(X_batch, y_batch, classes=[0, 1]
```

# Linear Classification

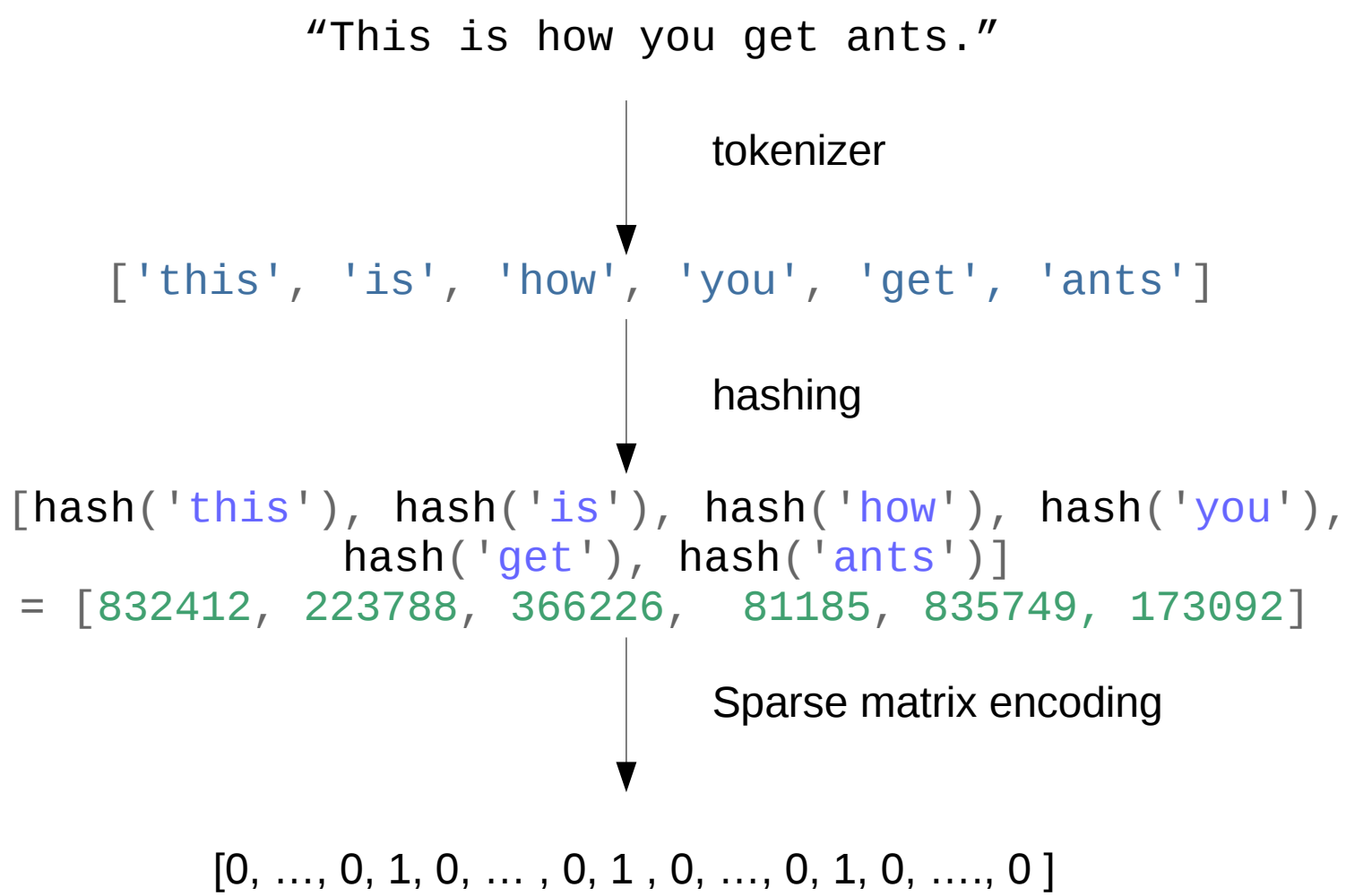
```
from sklearn.linear_model import SGDClassifier

sgd = SGDClassifier()

for i in range(n_iter):
    for batch_name in glob("*.pickle"):
        with open(batch_name) as f:
            X_batch, y_batch = pickle.load(batch_name)
            sgd.partial_fit(X_batch, y_batch, classes=[0, 1])
```

# 1st nonlinear option: Stateless Transformers

# Text Classification: Bag Of Word



# Text Classification: Hashing Trick



# Text Classification: Hashing Trick

```
sgd = SGDClassifier()
hashing_vectorizer = HashingVectorizer()

for batch_name in glob("*.pickle"):
    with open(batch_name) as f:
        text_batch, y_batch = pickle.load(batch_name)

    X_batch = hashing_vectorizer.transform(text_batch)
    sgd.partial_fit(X_batch, y_batch, classes=[0, 1]
```

# Kernel Approximation

```
sgd = SGDClassifier()
kernel_approximation = RBFSampler(gamma=.001, n_components=400)
kernel_approximation.fit(np.zeros(1, n_features))

for batch_name in glob("*.pickle"):
    with open(batch_name) as f:
        X_batch, y_batch = pickle.load(batch_name)
    X_kernel = kernel_approximation.transform(X_batch)
    sgd.partial_fit(X_kernel, y_batch, classes=[0, 1])
```

# Random Neural Nets

```
sgd = SGDClassifier()
random_basis = RandomBasisFunctions()
random_basis.fit(np.zeros(1, n_features))

for batch_name in glob("*.pickle"):
    with open(batch_name) as f:
        X_batch, y_batch = pickle.load(batch_name)
    X_random = random_basis.transform(X_batch)
    sgd.partial_fit(X_random, y_batch, classes=[0, 1])
```

(not merged yet)

# $2^{nd}$ nonlinear option:
# Learn Transformations on Subsets

# RandomForests

```
from sklearn.ensemble import RandomForestClassifier

X, y = load_my_subset_that_fits_in_ram()
rf = RandomForestClassifier(max_depth=5, n_estimators=100).fit(X, y)

rf_enc = OneHotEncoder()
rf_enc.fit(rf.apply(X))

sgd = SGDClassifier()

for batch_name in glob("*.pickle"):
    with open(batch_name) as f:
        X_batch, y_batch = pickle.load(batch_name)
    X_transformed = rf_enc.transform((rf.apply(X_batch)))
    sgd.partial_fit(X_transformed, y_batch, classes=[0, 1])
```

# 3rd nonlinear option:
# Online Nonlinear Classification

# Neural Networks (MLPs)

```
sgd = SGDClassifier()

for batch_name in glob("*.pickle"):
    with open(batch_name) as f:
        X_batch, y_batch = pickle.load(batch_name)
        sgd.partial_fit(X_batch, y_batch, classes=[0, 1])
```

(not merged yet)

# Neural Networks (MLPs)

```
nn = MLPClassifier(n_hidden=(1000, 1000))

for batch_name in glob("*.pickle"):
    with open(batch_name) as f:
        X_batch, y_batch = pickle.load(batch_name)
        nn.partial_fit(X_batch, y_batch, classes=[0, 1])
```

(not merged yet)

# Other algorithms

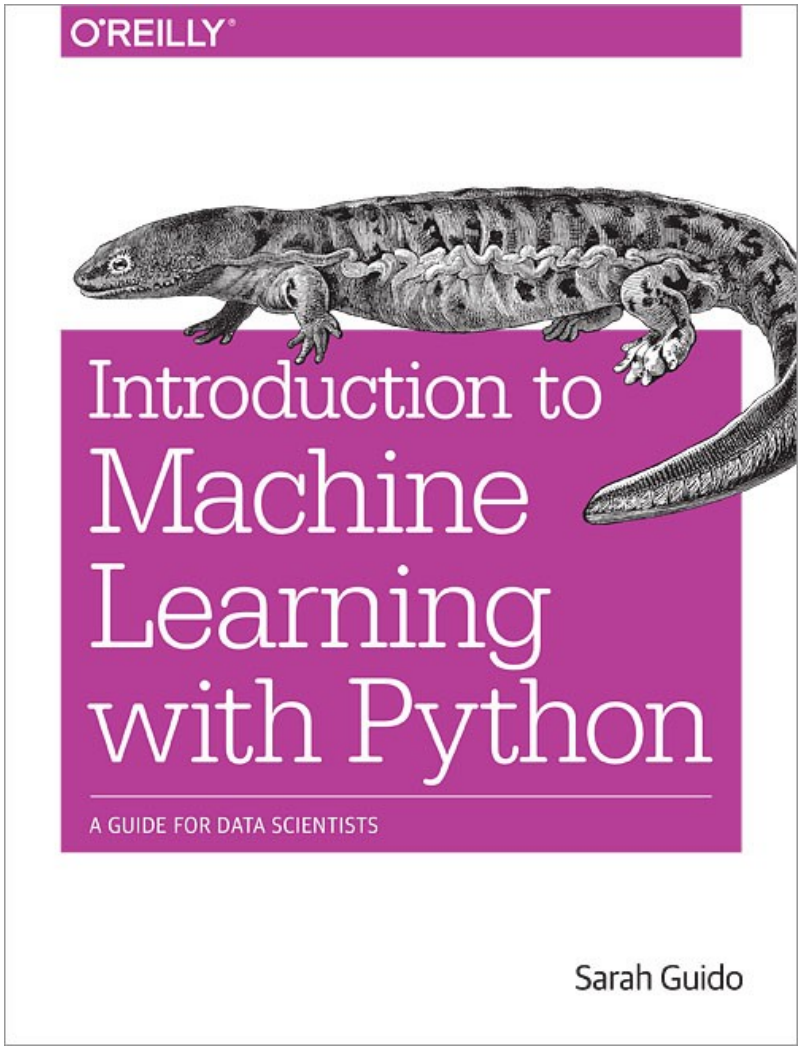
- Naive Bayes
- `MinibatchKMeans`
- `Birch`
- `IncrementalPCA`
- `MiniBatchDictionaryLearning`
- `Scalers`
- `...`

# What Else is Out There?

- Vowpal Wabbit (VW)
- More deep learning
- Hogwild!

# Video Series
# Advanced Machine Learning with scikit-learn

# Video Series
# Advanced Machine Learning with scikit-learn

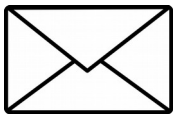


# Thank you!

(and talk to me if you still think you need a cluster for ML)

@amuellerml

@amueller



t3kcit@gmail.com